"Lower bounds on communication complexity in VLSI"

by

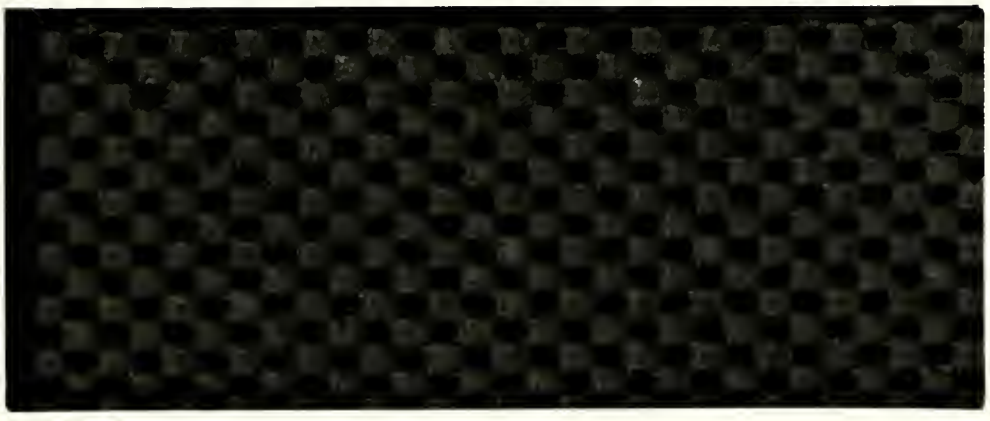
Richard Cole†
Alan Siegel‡

Ultracomputer Note #90
Technical Report #192
December, 1985

Ultracomputer Research Laboratory

NYU COMPSCI TR-192 c.1
Cole, Richard
Lower bounds on
communication complexity
in VLSI

New York University
Courant Institute of Mathematical Sciences
Division of Computer Science
251 Mercer Street, New York, NY 10012



"Lower bounds on communication complexity in VLSI"

by

Richard Cole†
Alan Siegel‡

Ultracomputer Note #90
Technical Report #192
December, 1985

† This work was supported in part by NSF Grant DCR-84-01633, and by an IBM Faculty Development Award.

‡ This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U. S. Department of Energy, under contract number DE-AC0276ER03077, and in part by ONR contract number ONR-N0014-85-K-0046.

ABSTRACT

We analyze a family of problems governing the VLSI complexity of broadcasting B bits of information in time T ($\leq B$) to each of N processors located along the perimeter of a two dimensional broadcast network. Among other results, we show:

- $AT^2 = \Theta(NB^2 \frac{\log N}{\log(2T/\log N)})$ for such network, when the output ports are unit distance apart. This bound seems to be the first AT^2 tradeoff which is not based on the number of bits that must cross a cut or boundary of some subcircuit.
- The above AT^2 bound scales linearly in the distance between adjacent ports, for ports distributed along the circuit perimeter with an approximately uniform spacing.
- Any vertex degree 3 graph, which has N vertices and breadth first search depth T, contains a complete binary tree of depth $\Omega(\frac{\log N}{\log(2T/\log N)})$.

These problems are of importance in the design and analysis of optimal VLSI circuits, such as sorters that have their input ports located along the circuit perimeter. Moreover, some of the proof techniques are of interest in their own right.

1. Introduction

This work analyzes the VLSI complexity of a basic function: the broadcast. Evidently, a broadcast circuit must, in general, contain a tree linking the output ports to the input port. Thus it is to be anticipated that the complexity of laying out a tree plays a central role in our results. It is well known that an N node, complete binary tree can be laid out in linear area (the H-tree layout). Most of the leaves of an H-tree, however, are located in the interior of the circuit, and not along the perimeter. In many applications – such as when the leaves represent output ports – the leaf nodes must be located exclusively along the perimeter of the circuit.

More specifically, we consider the following problem. Give a tight lower bound on the VLSI complexity of broadcasting B bits in time T to N sets of consecutive output ports, that is, for each bit b, there is a port in each set of output ports that receives bit b.

It should be emphasized that the intervals spanned by sets of output ports must be disjoint. This restriction prevents the problem from being solved, for instance, by B/T

disjoint broadcast trees, each of minimal size. Our formulation is chosen because it captures the problem of broadcasting to N processors (having one set of ports per processor), all of which are on chip in the VLSI model. Such a formulation arises in the problem of sorting, where it turns out that for suitable times and word lengths, the VLSI complexity is completely attributable to this broadcast complexity; the area and time required for the actual sorting turns out to be negligible.

Our VLSI model is, for the most part, the standard one used by [Thompson 79] to establish the first AT^2 tradeoffs for integrated circuits. The number of layers is fixed, at, say, two. The circuit (modulo its two layers) is two dimensional. Wires on a layer must be separated by unit distance, lest they shortcircuit. Wires incur a transmission delay of unity, independent of length. Input and output ports as well as all logic gates have unit sized dimensions and a unit time transmission delay. Logic gates are required to have an indegree and outdegree of at most two. More information about this model can be found in [Ullman 84].

In view of the application, it is appropriate to allow each bit variable to be read at repeated times and at many input ports. This generalization, it turns out, does not degrade the resulting bounds since all input ports are located within a single interval of consecutive ports. In order to get tight lower bounds, we are obliged to introduce one nonstandard simplification: the flow of information within the network is assumed to be conservative. In a conservative network, a gate cannot perform boolean algebra on its input bits. It can simply output a subset of its inputs. We could also allow unbounded storage of inputs, and require that the bits output be specific values that were previously input. In a conservative network, therefore, any bit traveling on any wire can be identified with the input variable from which the value originated. The issue of conservative versus nonconservative information flow is discussed further in Section 5.

Our lower bounds for the broadcast problem are a consequence of lower bounds for the following tree layout problem, which is of interest in its own right. We are given a set of N leaves, distributed along a straight line; the distance between the farthest leaves is S (which may be much larger than N). These leaves are to be connected by a binary tree of depth T. Give a tight lower bound on the area used by a rectangular circuit containing such a tree. We prove the tight lower bound $\Omega(\frac{S\log N}{\log(2T/\log N)})$. [Yao 81] stated the result for the case $S = N$; no proof was given. We also generalize the bound to the case that the distance between consecutive nodes is S/N, and remove any requirement that the circuit be rectangular, or convex. This is a substantial generalization, since the statement of Yao does not bound the circuit area, but rather the area of the convex hull. To achieve our bounds for broadcast networks, we need to show the stronger result that the total wire length W of the circuit satisfies $W = \Theta(\frac{S\log N}{\log(2T/\log N)})$. The conservative flow assumption is not needed for these results about trees; rather, it is used to ascertain the existence of disjoint paths of information flow, whose VLSI complexities can then be measured via our results for trees. The matching upper bound for our result is given in [Cole and Siegel 85].

The motivation for this work arose in the study of VLSI lower bounds for *perimeter* sorters. A perimeter sorter has its I/O ports on the perimeter. For circuits that sort N numbers lying in the range $[0,N^2]$, perimeter sorters turn out to be equivalent in size to *dense* sorters, which have no restrictions on the locations of their I/O ports. Optimal constructions for this number range (which actually covers $[0,N^{1+c}]$, for fixed $c>0$) originate with [Bilardi and Preparata 84] and [Leighton 84]. Lower bounds for circuits with perimeter I/O originate with [Yao 81], and the first AT^2 bounds for perimeter sorters occur in [Ullman 84] for the range $[0,N]$, and Leighton for the range $[0,N^7]$.

[Cole and Siegel 85, Siegel 85] solved the problem of optimal sorting for N numbers in the range $[1,M]$ in general. [Cole and Siegel 85] provided upper bounds, while general AT^2 lower bounds were addressed in [Siegel 85]. [Cole and Siegel 85] gives sorter constructions

that are tight for for essentially all number ranges for dense sorters, and that are tight in many instances for perimeter sorters. These constructions include circuits that match AT^2 and AT complexity tradeoffs, depending on time, area, and word length. Lower bounds on the AT complexity for dense sorters originate with [Bilardi and Preparata 85].

For perimeter sorters, some of the lower bounds depend critically on the lower bounds for the VLSI complexity of broadcasting. This is in contrast with the complete spectrum of bounds for dense sorting circuits, where it suffices to consider information flow arguments in general, and, in certain cases, pipeline delays as well [Siegel 85], [Bilardi and Preparata 85]. Such arguments are inadequate to prove lower bounds on the area needed for broadcasts, since the information flow is too small. A different approach is required. In Section 5 we describe more precisely the communication problems and we show how to reduce them to the tree problem defined above.

In Section 2, we consider the case $T = \log N$, and obtain a lower bound of $\Omega(\log N)$ on the layout height of a circuit that contains a complete binary tree of depth $\log N$, with the leaves along one side of the circuit. This result holds regardless of the distribution of the leaves. It is based on the wire length bound of [Brent and Kung 80] for such trees.

In Section 3, we prove that any binary tree, of height T, with N leaves, must contain a complete binary tree of depth $\Omega(\frac{\log N}{\log(2T/\log N}$). By the result of Section 2, we deduce an $\Omega(\frac{\log N}{\log(2T/\log N)})$ lower bound on the layout height of a circuit containing this complete binary tree, which is therefore also a lower bound on the layout height of a circuit containing the binary tree of depth T and N leaves. Interestingly, this lower bound is obtained by computing an upper bound. In particular, the result follows from estimating the maximum number of leaves in a depth T binary tree that contains no complete binary tree of depth M.

In Section 4, we obtain a lower bound on the area used by a depth T binary tree having N leaves distributed with uniform spacing along a line segment. The lower bound is obtained

by estimating the horizontal wire length used by such a tree.

2. log N depth trees

In this section, we let $T = \log N$. [Brent and Kung 80] proved the following.

Lemma 1 [Brent and Kung]: Let G be a VLSI layout of a complete N-leaf binary tree, where the leaves are located along a straight line. Then the total wire length of G is $\Omega(N \log N)$.

For completeness, we give a proof which follows the basic idea of [Brent and Kung 80], but is more direct.

Proof: Let the line be horizontal, and let the distance between two points be the horizontal displacement: $dist[(x,y),(u,v)] = |u-x|$. If P is a path with endpoints p_0 and p_1, we define $||P|| = dist[p_0,p_1]$. Consider the path P between the root and the most distant leaf; evidently $||P|| \geq N/2$. Let $v_0, v_1, v_2, \ldots, v_{\log N}$ be the vertices on this path, where v_0 is the root and $v_{\log N}$ is the leaf. Delete P, and the wire edges descending from each node along P. We are left with disjoint complete binary trees, containing, respectively, $N/2$, $N/4$, ...,1 leaves. Thus we obtain the following recurrence equation for the total horizontal wire length, $L(N)$:

$$L(N) \geq N/2 + \sum_{i=1}^{\log N} L(N/2^i) \quad \text{for } N > 1$$
$$L(1) = 0.$$

This has the solution $L(N) = \Omega(N \log N)$. □

It follows, of course, that if adjacent leaves have a spacing of unity, that the layout height of the circuit is $\Omega(\log N)$. For applications involving broadcasting, a leaf might represent a processing element having large length and width. Thus it is necessary to know how the area and layout height are affected when we permit the leaves to be spread out with a spacing that is $>> 1$.

Corollary 2: Consider a circuit that contains a complete binary tree of N leaves, which are constrained to lie along a straight line, and for which some pair of leaves are distance S

apart. Then any layout of the circuit which is convex, or which has the leaves evenly spaced along the line uses area $\Omega(S \log N)$.

Proof: We normalize the distance measure so that every pair of adjacent ports is separated by unity. Formally, let l_i be the x coordinate of the i-th leaf. We define horizontal distance by the Riemann-Stieltjes measure $d\mu = \frac{dx}{l_{i+1} - l_i}$, for $x \in [l_i, l_{1+1}]$; area is defined by $dy\,d\mu$. Applying Lemma 1 gives an $\Omega(N \log N)$ bound for (normalized) area, whence the layout height must again be $\Omega(\log N)$. Furthermore, if the spacing between adjacent leaves does not vary, then the true area of the layout is $\Omega(L \log N)$. □

Remark: The proof of Corollary 2 actually shows that a bounded degree graph of N nodes can be laid out in a box of $O(N)$ width so that its layout height is $O(C)$, where C is the crossing number of the graph. The crossing number is defined to be the minimum, over all layouts, of the maximum number of edges and vertices crossing any vertical line through the layout. It also follows that the lower bound on layout height does not depend on the fact that the leaves are on a straight line.

3. The general bound on layout height

In this and the following section, we generalize the results of Section 2 to N-leaf binary trees of depth $T >> \log N$. [Yao 81] states the following:

Theorem 3 [Yao]: Let H be a layout for a binary tree that has depth T, and that has N leaves located along a straight line. Then the convex hull of H has area $\Omega(\frac{N \log N}{\log(2T/\log N)})$.

We prove a more general result which immediately implies this theorem. In Section 4, we strengthen this bound by removing the convexity requirement and by generalizing the bound to include the possibility that the leaves are spread out along a large distance. Finally, Section 5 shows how to apply these stronger results to attain lower bounds on the area-time tradeoff for broadcasting in a conservative network.

We consider the following problem: Give an upper bound, $N(T,M)$, on the number of leaves in a depth-T binary tree H, which contains no complete binary tree having more than 2^M leaves. (Tree H is said to contain tree K if there is a mapping of the vertices of K into the vertices of H that preserves the ancestor relationships.)

Thus, given a binary tree H, which has depth T and which has N leaves that lie on a straight line, we can deduce a lower bound on the depth (M) of the largest complete binary tree contained in H. By Corollary 2, we deduce that the layout height of a circuit containing a depth M complete binary tree is $\Omega(M)$. Clearly, this is also a lower bound on the layout height for a circuit containing H.

Theorem 4: Let H be a binary tree that has depth T and N leaves. Then H contains a complete binary tree of depth $\Omega(\frac{\log N}{\log(2T/\log N)})$.

Proof: We state and solve a recurrence equation for $N(T,M)$.

$$N(T,M) = N(T-1,M) + N(T-1,M-1) \quad \text{if } T>M>0.$$

$$N(T,M) = 2N(T-1,M-1) \quad \text{if } T=M>0.$$

$$\text{and } N(T,0) = 1.$$

Clearly, $N(T,T) = 2^T$.

Define $h = T-M$, and write $S(h,M) = N(T,M)$. Then

$$S(h,M) = S(h-1,M) + S(h,M-1) \quad h\geq 1, M\geq 1.$$

$$S(0,M) = 2^M.$$

$$S(h,0) = 1.$$

We solve this recurrence equation using the following generating function.

$$g(x,y) = \sum_{h\geq 0, M\geq 0} S(h,M)x^h y^M.$$

This has the solution

$$g(x,y) = (1-x-y)^{-1}(1-y)(1-2y)^{-1}.$$

We deduce

$$S(h,M) = \sum_{r=0}^{M} \binom{h+r}{h} 2^{M-r} - \sum_{r=0}^{M-1} \binom{h+r}{h} 2^{M-r-1} \leq \sum_{r=0}^{M} \binom{h+r}{h} 2^{M-r}$$

There are two cases to consider: $h > M$ and $h \leq M$.

Case 1: $h > M$.

Here $2M < T < 2h$. Also, we assume $M \geq 1$.

The largest term in the sum is $\binom{h+M}{h}$.

So $N = S(h,M) \leq (M+1)\binom{h+M}{h}$.

By Stirling's formula ($p < n! < p(1+1/(4n))$, where $p = \sqrt{2\pi} n^{n+1/2} e^{-n}$); we obtain

$$\log N \leq \log M + h\log(1+M/h) + M\log(1+h/M) \leq M + M + M\log T/M \leq 3M\log(T/M).$$

Thus $M \geq \dfrac{\log N}{3\log(T/M)} \geq \dfrac{\log N}{3\log(T/\log N) + 3\log\log(T/M) + 3\log 3} \geq \dfrac{\log N}{12\log(2T/\log N)}$.

Hence $M = \Omega(\dfrac{\log N}{\log(2T/\log N)})$.

Case 2: $h \leq M$.

Here $2h \leq T \leq 2M$.

The largest term in the sum is $\binom{2h}{h} 2^{M-h}$.

So $N = S(h,M) \leq (M+1)\binom{2h}{h} 2^{M-h}$.

We deduce that $\log N \leq \log(M+1) + M + h \leq 3M$; so $M \geq \dfrac{\log N}{3} \geq \dfrac{\log N}{3\log(2T/\log N)}$.

Thus, in all cases, $M = \Omega(\dfrac{\log N}{\log(2T/\log N)})$.

We deduce the layout height of the circuit is $\Omega(M) = \Omega(\dfrac{\log N}{\log(2T/\log N)})$. □

Corollary 5: Let G be a degree 3 graph that has breadth first search (BFS) depth T and that contains a set of N distinguished nodes. Then G contains a complete binary tree that has all its leaves in the distinguished set and that has depth $\Omega(\frac{\log N}{\log(2T/\log N)})$.

Proof: Let H be a BFS tree of G. Recursively delete all leaves of G that are not distinguished. Recursively delete the root while it has outdegree 1 and is not distinguished. Replace all maximal chains of undistinguished nodes that have outdegree 1 by simple edges. Add two leaf children to each (distinguished) leaf. Add a single leaf child to each (distinguished) node with outdegree 1. We apply Theorem 4 to H, and remove the leaves of the resulting binary tree. □

4. Wire length

We now prove a lower bound on the horizontal wire length used to layout a depth T binary tree which has N leaves located with uniform spacing along a horizontal line. The results in Section 3 are not sufficient to bound wire length properly. Our new bound is necessary for Section 5, where we need to bound the layout area for a forest of such trees, which are used to broadcast a message of $B >> T$ bits.

We begin by establishing some preliminary lemmas about the layout structure of a binary tree that has its N leaves located on a straight line L, and that has minimal horizontal wire length. The following lemmas all refer to such a tree, unless otherwise specified. The lemmas show that the layout obtained by following a greedy inductive strategy is a layout using minimum horizontal wire length. Actually, we have defined the horizontal wire length of an edge to be the horizontal distance between its two endpoints; in general, this measure underestimates the horizontal wire length since an edge need not be monotone in x, but, in fact, it yields the correct value for the layout we show to be optimal. When we refer to a binary tree using minimal horizontal wire length, we intend the definition of horizontal wire length given above.

Lemma 6: Let v be an internal node of the binary tree and let C_v be the horizontal interval spanned by its children. Then v (i.e., the orthogonal projection of v onto L) lies in C_v.

Proof: See Appendix. □

Lemma 7: For each internal node v of the binary tree, let I_v be the horizontal interval spanned by the leaves of the subtree rooted at v. Let u and v be two unrelated nodes (neither is an ancestor of the other). Then I_u and I_v are disjoint.

Proof: See Appendix. □

Lemma 8: Let v be an internal node of the binary tree. Then v is aligned vertically with one of its children.

Proof: See Appendix. □

Definitions: Let vertex v have two children: v is connected to one child, the *vertical* child, by a vertical wire and to the other child, the *horizontal* child, by an edge with a non-zero horizontal length; this edge is called the horizontal exiting edge for v. We define the subtree rooted at the vertical (resp. horizontal) child to be the *vertical* (resp. *horizontal*) subtree of v. Let v be a vertex in the binary tree. A left (resp. right) spanning edge for v is an edge whose horizontal extent includes the portion of I_v to the left (resp. right) of v, and that extends beyond I_v to the left (resp. right).

Lemma 9: Let w be an ancestor of v in the binary tree. Let e be the horizontal edge exiting w. If e (i.e. the projection of e onto L) overlaps with I_v then e is a spanning edge for v. Also, each spanning edge for v is a horizontal edge that exits some ancestor of v.

Proof: See Appendix. □

Lemma 10: Let e be the horizontal edge exiting vertex w, if any. Suppose w is unrelated to vertex v. Then e and I_v do not overlap.

Proof: See Appendix. □

Lemma 11: Let w be a vertex in the binary tree. If w has more right (resp. left) spanning edges than left (resp. right), then the exiting horizontal edge for w, if any, exits to the left (resp. right). In fact, there is at most a difference of one between the numbers of left and right spanning edges for any vertex w.

Proof: See Appendix. □

Corollary: In each vertical chain of vertices the horizontal edges can be chosen to exit alternatingly left and right.

Definition: The cost of a node is defined as follows. The cost of the root is zero. The cost of a vertical child is zero. The cost of a horizontal child is the number of spanning edges of the child, including the edge connecting it to its parent. The cost of a tree is the largest cost of all the nodes (not just leaves) in the tree.

The intuition behind this definition of cost is quite simple. Consider a node that is not directly below its parent. Its presence in the graph causes a unit of horizontal length in the edge from its parent, and in all of the spanning edges it shares with its parent, i.e. the edges which pass over the node. As for a node that lies below its parent, its addition to the tree causes no increase in the horizontal edge length. The horizontal edge length in a tree is, evidently, at least at the sum of the node costs.

Lemma 12: Let $M+1$ be the largest cost of any leaf. Consider a leaf, v, to which the path depth from the root is less than T; it has M or $M+1$ left spanning edges and M or $M+1$ right spanning edges. Thus the cost of adding a horizontal child to this leaf is at least $M+1$.

Proof: If v had fewer than M spanning edges to one side, the left side say, then we could remove a leaf of cost $M+1$, and add a pair of children to v. The cost of (and the increment in horizontal edge length due to) the horizontal child of v would be at most M, less than the cost of the removed leaf, while the vertical child causes no change in the horizontal edge length. The depth of the tree would still be T, but the total horizontal edge length would be decreased. □

Definition: A tree of cost M is said to be *full* if every leaf of depth $<T$ has M spanning edges to both the left and the right.

It follows from Lemma 12 that the following construction produces a tree of minimal horizontal wire length. In turn, lay out a full tree of cost 0, a full tree of cost 1, a full tree of cost 2, ..., a full tree of cost M, and then some additional leaves, each of cost 0 or $M+1$. The full tree of cost $i+1$ is obtained from the full tree of cost i as follows. Choose some leaf, v, in the tree we have constructed so far (initially, the full tree of cost i) that has i spanning edges to one side and depth $<T$. Provide v with two children, connecting them as follows. One child is vertically aligned with v, and the horizontal edge exiting v is connected to the other child; this edge exits to the left or right according to the rules given by Lemma 11 and the corollary following it. Room is made for the horizontal child by cutting the graph above the child, and horizontally shifting a side of the graph by unit distance, thus incrementing by 1 the length of each of the the child's spanning edges. We will have obtained a full tree of cost $i+1$ when there are no leaves to which this construction can be applied. The final addition of leaves is done in exactly the same way; the only difference is that we may stop before obtaining a full tree of cost $M+1$.

Let $P(T,M)$ denote the maximum number of leaves in a full tree of cost M, where the root has no spanning edges. Let $\tilde{P}(T,M)$ denote the number of leaves in a full tree of cost M, where the root has one left spanning edge (or equivalently, one right spanning edge). Let $K(T,M)$ denote the horizontal wire length needed to lay out a full tree of cost M if the root has no spanning edges, and let $\tilde{K}(T,M)$ denote the horizontal wire length needed to lay out a full tree of cost M if the root has one left (or right) spanning edge. Let $H(T,N)$ denote the minimum horizontal wire length needed to lay out a binary tree of N leaves, of depth T, assuming the root has no spanning edges, and let $\tilde{H}(T,N)$ denote the minimum horizontal wire length needed to lay out the same tree, except that the root has one left (or right) spanning edge. By Lemma 12 and the discussion following it we deduce

Lemma 13: Let $M > 0$, and suppose that $P(T,M) \leq N \leq P(T,M+1)$, $\tilde{P}(T,M) \leq N \leq \tilde{P}(T,M+1)$, as appropriate. Then

$$H(T,N) = K(T,M) + (M+1)[N - P(T,M)]$$

$$\tilde{H}(T,N) = \tilde{K}(T,M) + (M+1)[N - \tilde{P}(T,M)]$$

$$K(T,M+1) = K(T,M) + (M+1)[P(T,M+1) - P(T,M)]$$

$$\tilde{K}(T,M+1) = \tilde{K}(T,M) + (M+1)[\tilde{P}(T,M+1) - \tilde{P}(T,M)].$$

Furthermore,

$$H(T,N) \geq [N - P(T,\lfloor M/4 \rfloor)]M/4,$$

$$\tilde{H}(T,N) \geq [N - \tilde{P}(T,\lfloor M/4 \rfloor)]M/4.$$

Proof: Trivial. □

Recall the function $N(T,M)$ from Section 3. It is easy to show that

$$P(T,M) \leq N(T,2M)$$

$$\tilde{P}(T,M) \leq N(T,2M-1).$$

Lemma 14: $N(T,M) - N(T,\lfloor M/2 \rfloor) \geq 1/2N(T,M)$, for $M \geq 2$.

Proof: See Appendix. □

We deduce that $H(T,N), \tilde{H}(T,N) \geq [N - 1/2N(T,M)]M/4 = \Omega(NM)$. That is, the minimum horizontal wire length for such a circuit is $\Omega(N \frac{\log N}{\log(2T/\log N)})$, using the lower bound on M from Section 3.

Remark: We could have obtained the result directly by solving a recurrence equation for $\tilde{P}$. However, the expression for $\tilde{P}$ is much more complicated than that for N, thus making it more desirable to obtain the result indirectly.

We have proved

Theorem 15: A binary tree that has depth T, and N leaves, which are evenly distributed along a horizontal line segment of length S, has total wire length $\Omega(S \frac{\log N}{\log(2T/\log N)})$. □

5. Conservative flow versus non-conservative flow.

We want to obtain the following result. We are given a series of disjoint intervals on a straight line, each of length $\geq B/T$. Call the leftmost interval the input interval, and the others output intervals. Each interval contains at least B/T ports. We want to provide a (tight) lower bound for the area required by a circuit that broadcasts B bits from the input interval to all the output intervals (that is, each bit is sent out from some, or several, input ports, and is received by at least one input port in each interval). If we assume that the circuit is conservative (that is, each node in the circuit just forwards the bits it receives without changing them) then it is easy to reduce this to the tree problem considered in this paper, as we show. Without the conservative flow assumption, unless $B \leq T$, we are unable to prove a tight lower bound. Proving the lower bound, in general, without the conservative flow assumption, is an interesting open problem. It is but one instance of the general problem of comparing conservative flow to non-conservative flow [Aggarwal 85]. For example, [Aggarwal 84] considers an unrelated problem concerning the VLSI complexity of communication on X-trees: he obtained a tight lower bound under a conservative flow assumption. [Siegel 85b] provided a tight lower bound for this problem, without the conservative flow assumption. Yet for the most part, the relationship between conservative and general information flow seems to be quite fundamental but as yet poorly understood [Chandra 85], [Kosaraju 85].

We now describe the reduction of the broadcast problem to the tree problem, under the conservative flow assumption. Consider the two sets of output intervals obtained by selecting alternate intervals. We discard the set of greater length, that is, we require the broadcast to be made solely to the intervals in the non-discarded set. Next we normalize the horizontal lengths: the length of each discarded interval is normalized to one, and the length of each remaining interval is normalized to zero. The normalized length of the network is $N/2$. Consider a single bit of the B bits to be broadcast. It is broadcast on some subnetwork.

Perform a breadth first search of this subnetwork, starting from the input ports. The search yields a ternary tree of depth $\leq T$ (since, in the VLSI model, each node can have at most 4 exiting wires), with the root in the input interval and a leaf in each output interval (actually, the subnetwork may comprise several trees, each of depth $\leq T$, each with a root in the input interval, such that each output interval has a leaf in at least one tree). In Lemma 18, in the appendix, we show that if we double T, we can replace the ternary tree(s) by binary tree(s). If there is a single tree, it has normalized horizontal wire length $\Omega(\frac{N \log N}{\log(2T/\log N)})$, as shown in Section 4. In the next paragraph, we show that this bound also applies if there is more than one tree present. Each edge in the network can transmit at most T bits. By allocating $1/T$ of the length of an edge to each tree associated with a bit that travels along the edge, we deduce the overall normalized horizontal wire length is $\Omega(B/T \frac{N \log N}{\log(2T/\log N)})$. Since the normalized length of the circuit is $N/2$, we deduce that a rectangular circuit must have height $\Omega(B/T \frac{\log N}{\log(2T/\log N)})$. Hence a rectangular circuit of length S will need area

$$\Omega(B/T \frac{S \log N}{\log(2T/\log N)}).$$

Although the above analysis characterizes a single broadcast tree, it turns out that even a message of T bits, in this model, should be broadcast on several tree networks, to minimize the total horizontal wire length. It is easy to show, as in Section 4, that the intervals spanned by the leaves of each tree are disjoint, in the case of a minimal area network. Let $Z_i = \sum_{j=1}^{i-1} \bar{P}(i,T)$, for $i \leq k+1$, and let $N = Z_{k+1}$. Then the arguments of Section 4 can be used to show that the normalized horizontal wire length used by the i-th leftmost tree, for $i = 1,2, \ldots, k$, in this case, is $\bar{K}(i,M) + Z_i$ (where the input interval is the leftmost interval). It is then easy to deduce that the normalized horizontal wire length used by the circuit for any N is $\Omega(\frac{N \log N}{\log(2T/\log N)})$. (An alternative approach is to minimize the expression for the sum of the horizontal wire lengths of the trees using Jensen's inequality;

this yields the same lower bound. However, it does not indicate which is the best layout.)

We have shown

Theorem 16: A VLSI network that broadcasts B bits in time T ($B \geq T$) from one set of input ports to N sets of output ports, where the intervals spanned by each set of ports are disjoint, has the following complexity, under the conservative flow assumption:

(i) If the ports are uniformly distributed along a line of length Λ/T ($\geq NB/T$), then $AT^2 = \Omega(B \frac{\Lambda \log N}{\log(2T/\log N)})$.

(ii) If the ports are distributed along a line of length Λ/T ($\geq NB/T$), then for a convex circuit $AT^2 = \Omega(B \frac{\Lambda \log N}{\log(2T/\log N)})$. □

The following related problem also arises when proving lower bounds for perimeter sorters. Again, we have intervals as described above, but the input and output roles are reversed. In input interval i, the B bits $X_{i1}, X_{i2}, \ldots, X_{iB}$ are input. The B output bits are given by $\bigcirc_{i=1}^{N} X_{ij}$, $j = 1,2,\ldots,B$, where $\bigcirc$ is the Exclusive OR operation. Again, by an appropriate assumption, similar to conservative flow, we are able to obtain the same lower bounds as above. The assumption essentially states that the only computation that can be done is an Exclusive OR of bits that contribute to the same output bit. It is then easy to reduce this problem to the tree problem considered in Section 4: we use the same argument as for the broadcast problem. We obtain

Theorem 17: A VLSI network as defined in the last paragraph has the following complexity:

(i) If the ports are uniformly distributed along a line of length Λ/T ($\geq NB/T$), then $AT^2 = \Omega(B \frac{\Lambda \log N}{\log(2T/\log N)})$.

(ii) If the ports are distributed along a line of length Λ/T ($\geq NB/T$), then for a convex circuit $AT^2 = \Omega(B \frac{\Lambda \log N}{\log(2T/\log N)})$.

References

Aggarwal, A., [1984]. "A comparative study of X-tree, pyramid and related machines," *Proc. 25-th Annual Symposium on Foundations of Computer Science*, 89–99.

Aggarwal, A., [1985]. Private communication.

Bilardi, G. and F. Preparata, [1984]. "A Minimum Area VLSI Network for $O(\log n)$ Sorting," *Proc. Sixteenth Annual ACM Symposium on the Theory of Computing*, 64–70.

Bilardi, G. and F. Preparata, [1985]. "Tessellation Techniques for Area-Time Lower Bounds with Applications to Sorting," CISS, proceedings to appear.

Brent, R.P. and H.-T. Kung, [1980]. "On the area of binary tree layouts," *Information Processing Letters*, **11**:1, pp. 46–48.

Chandra, A., [1985]. Private communication.

Cole, R. and A. Siegel, [1985]. "On information flow and sorting: new upper and lower bounds for VLSI circuits," *Proc. 26-th Annual Symposium on Foundations of Computer Science*, 208–221. See also 'Optimal VLSI circuits for sorting," Courant Institute technical report.

Kosaraju, S.R., [1985]. Private communication.

Leighton, F.T., [1984]. "Tight Bounds on the Complexity of Parallel Sorting," *Proc. Sixteenth Annual ACM Symposium on the Theory of Computing*, 71–80.

Siegel, A., [1985]. "Computing information flow for lower bounds in VLSI circuits," Courant Institute technical report.

Siegel, A., [1985b]. —— in progress.

Thompson, C.D., [1979]. "Area Time Complexity for VLSI," *Proc. Eleventh Annual ACM Symposium on the Theory of Computing*, 81–88.

Ullman, J.D., [1984]. *Computational Aspects of VLSI*, Computer Science Press.

Yao, A., [1981]. "The entropic limitations on VLSI computations," *Proc. 13-th Annual ACM Symposium on Theory of Computing*, 308–311.

Appendix: Proofs of lemmas

Proof of Lemma 6: If not, we move v towards C_v; this reduces the horizontal wire length.

Proof of Lemma 7: The result is proved by induction. First, we need a definition. The *level* of a node is the number of edges on the path from the root to the node. The *height* of the tree is the maximum level of any node in the tree. The *height* of a node is the height of the subtree rooted at that node. The induction is on T, the maximum of the heights of u and v.

The base case $T = 0$ is trivial, for then the tree consists of just one node. We prove the inductive step in two parts. First, suppose that u and v both have height $T+1$. Let w and z be the children of u, x and y the children of v. We want to show that I_u and I_v are disjoint. By the inductive hypothesis, I_w, I_z, I_x, and I_y are pairwise disjoint. WLOG suppose that u is to the left of v. If w and z are not the two leftmost vertices among $\{w,z,x,y\}$, the horizontal wire length is reduced by making u the parent of the two leftmost vertices among $\{w,z,x,y\}$, and v the parent of the other two vertices. (Recall that by Lemma 6, u (resp. v) is vertically aligned with a point in $C_u \subseteq I_u$ (resp. $C_v \subseteq I_v$).) This change reduces the total horizontal wire length, while keeping the level of the leaves unaltered.

The second part handles the case that u has height $T+1$ and v has height $< T+1$. In this case we seek a pair of nodes, v' and v'', ancestors of v (possibly $v = v'$), such that v' has height $< T+1$ and v'' has height $\geq T+1$. We prove that $I_{v'}$ and I_u are disjoint; thus I_v and I_u are also disjoint.

We apply the inductive hypothesis to each of w and v', z and v'; we deduce that I_w and $I_{v'}$, I_z and $I_{v'}$ are disjoint. Hence a problem arises only if $I_{v'}$ lies between I_w and I_z; we show that this cannot happen.

In the event that v'' has height $T+1$ we apply the argument from the first part to v'' and u to show that $I_{v''}$ and I_u are disjoint (hence I_u and I_v are disjoint in this case). The only remaining case occurs if v'' has height $> T+1$ and v' has height $< T+1$. WLOG let v' be the

left child of v''. Let z be the right child of u. We interchange z and v'. The interchange does not increase the height of either u or v''. Next, if necessary, we shift u and v'' to ensure that the horizontal wire length does not increase as a result of the interchange. Let t' be the original sibling of v'. If u and v'' occur in left to right order the interchange reduced the horizontal wire length. If u and v'' occur in right to left order, we move u to the left, until it is aligned with v', and we move v'' to the right, until it is aligned with the leftmost of z and t'; this ensures that the net change in the horizontal wire length is a decrease.

Proof of Lemma 8: Suppose (for a contradiction) that a node v is not vertically aligned with either child. We show that it must be vertically aligned with its parent. If not, moving v (horizontally) towards its parent reduces the total horizontal wire length. If we cannot move v closer to its parent this is because w, its sibling, is in the way. But, by Lemma 7, I_v and I_w are disjoint, and w is above I_w. Hence v must be at an endpoint of I_v. Since, by Lemma 6, $C_v \subseteq I_v$ we deduce that v is vertically aligned with one of its children. Thus, if v is not vertically aligned with one of its children, then it is vertically aligned with its parent.

Next, we show that, in fact, v is vertically aligned with one of its children. Consider the longest sequence of ancestors of v, including its parent, vertically aligned with v. Moveing this sequence of ancestors, including v, to one of the left or right, does not increase the total horizontal wire length. Thus we can move this sequence of vertices until v is aligned with one of its children, or the motion is blocked. The motion is blocked only when a vertex tries to move to a location already occupied by its sibling. But, by an argument similar to the one used in the previous paragraph, v in then vertically aligned with one of its children.

Proof of Lemma 9: Let e be a left spanning edge for v. If e exits vertex w, then I_w and I_v overlap, since the span of e is contained in I_w. Hence, by Lemma 7, one of w and v is an ancestor of the other. But since e extends beyond I_v (by definition) we deduce that w is an ancestor of v.

Conversely, suppose that e is a horizontal edge, exiting w, overlapping I_v to the left of v, where v is a descendant of w. There are two cases to consider: either v is in the vertical subtree of w, or v is in the horizontal subtree of w. We handle the former case first. Let u be the vertical child of w, and let x be the horizontal child of w (possibly $u = v$). Then, by Lemma 7, since I_x and I_u are disjoint, and since e extends to x, which is contained in I_x, we deduce that e extends beyond I_u, and hence it extends beyond I_v also. Next, we show that if e overlaps with I_v then v and e are on the same side of w. For if v was on the opposite side of w, then there would be a vertex z unrelated to v, vertically aligned with w. As I_z and I_v are disjoint (Lemma 7) we deduce that e and I_v are also disjoint. But we assumed e and I_v overlap; also e reaches from w to I_x. Thus e must be a left spanning edge for v.

In the latter case (using the same notation) x is an ancestor of v. By an argument similar to that used in the last paragraph, if v and e are on opposite sides of x then e and I_v do not overlap. Thus v and e are on the same side of x. If v is vertically aligned with x then e is a right spanning edge for v. If v is not vertically aligned with x then there is a vertex z, unrelated to v, vertically aligned with x. I_z and I_v are disjoint, as are I_u and I_v. But e reaches from I_z to I_u; thus $I_v \subseteq e$. That is, e is both a left and a right spanning edge for v.

Proof of Lemma 10: e is contained in I_w. By Lemma 7, I_w and I_v do not overlap.

Proof of Lemma 11: We prove the result by induction on the height of w. It is clearly true for nodes of height 0 and 1. So consider a node w of height $h+1 \geq 2$. Let v and u be, respectively, the vertical and horizontal children of w. let a and b, c and d, be the children of u and v, respectively, with a and d being the vertical children. (See figure 1).

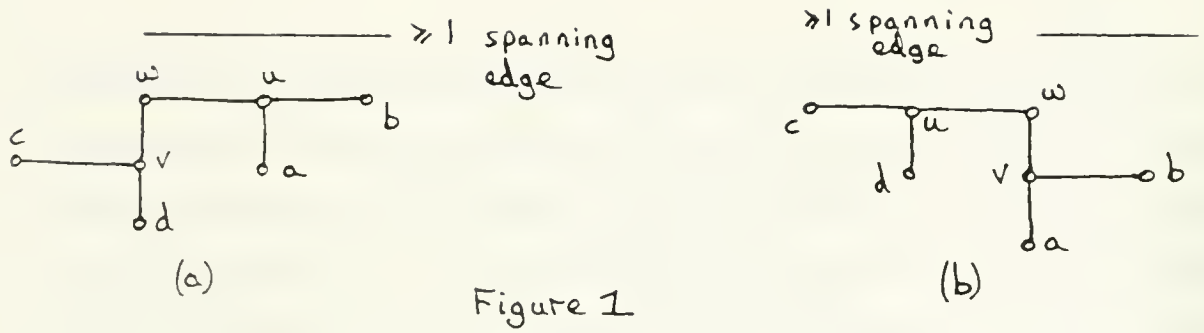


Figure 1

The layout of a, b, c and d can be deduced by applying the inductive hypothesis. Suppose that u is to the right of w. Then rearrange the nodes as illustrated in figure 1b; this reduces the total horizontal wire length. Hence we can deduce that u is to the left of w; that is, the edge exiting w horizontally exits to the left.

We prove the second claim by induction on the level of a node in a given tree. The node at level 0, the root, has no spanning edges (on occasion, it will be convenient to allow the root to have one left, or right, spanning edge). Thus the result is true for the base case. The inductive step is proved as follows. Suppose the result is true for node w, and we want to prove it for node v, a child of w. If v is the vertical child, then the spanning edges for v are comprised by the spanning edges for w plus the edge exiting w. Using the first claim, we deduce the second claim is true for v, in this case. If v is the horizontal child, WLOG assume v is to the left of w. The left spanning edges for w are both left and right spanning edges for v, while the right spanning edges for w, that are not left spanning edges for w, are not spanning edges for v; also, the edge exiting w (to the left) is a right spanning edge for v. It is clear that the second claim holds for v in this case also.

Proof of Lemma 14: For $M = 1,2$, the result can easily be checked. For $M > 2$ we show that $N(T,M) - N(T,\lfloor M/2 \rfloor) \geq N(T,\lfloor M/2 \rfloor)$.

$$N(T,M) - N(T,\lfloor M/2 \rfloor) \geq \sum_{r=0}^{M} \binom{h+r}{h} 2^{M-r-1} - \sum_{r=0}^{\lfloor M/2 \rfloor} \binom{h+r}{h} 2^{\lfloor M/2 \rfloor - r}$$

$$\geq \sum_{r=0}^{\lfloor M/2 \rfloor} \binom{h+r}{h} 2^{\lfloor M/2 \rfloor - r} \left(2^{\lceil M/2 \rceil - 1} - 1\right)$$

$$\geq \sum_{r=0}^{\lfloor M/2 \rfloor} \binom{h+r}{h} 2^{\lfloor M/2 \rfloor - r} \geq N(T, \lfloor M/2 \rfloor).$$

Lemma 18: Given a VLSI layout of a ternary tree of depth T, there exists a VLSI layout of a binary tree with the same leaves, having depth at most $2T$, using the same horizontal wire length.

Proof: We double the number of horizontal levels in the circuit. We lay out the circuit as before except that we only use alternate horizontal levels for the horizontal wires and we double the length of all vertical wires. For each vertex of degree 4 we perform the change illustrated in figure 2. This leaves the horizontal wire length unchanged, at most doubles the depth of the tree, and reduces the degree of each vertex to at most 3.

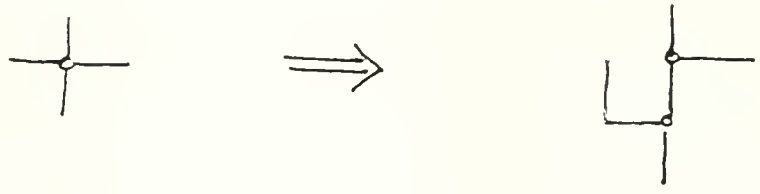


Figure 2

This book may be kept JUL 2 o 1986
FOURTEEN DAYS
A fine will be charged for each day the book is kept overtime.

GAYLORD 142 PRINTED IN U S A

NYU COMPSCI TR-192 c.1
Cole, Richard
Lower bounds on
communication complexity
in VLSI

NYU COMPSCI TR-192 c.1
Cole, Richard
Lower bounds on
communication complexity
in VLSI

LIBRARY
N.Y.U. Courant Institute of
Mathematical Sciences
251 Mercer St.
New York, N. Y. 10012